Agenda

- Critical Purpose of Logs - Relational Database Rules
- Database Transactions Require Logs
- Informix Database Logging Modes
- How the Physical Log Works
- How the Logical Logs Work
- Monitoring Logical Logs Status
- Logical Logs Problems
- Informix Checkpoints and Logs
- Informix Fast Recovery
- Backing Up Logical Logs Automatically
- Managing Logs Using Onparams and InformixHQ

Relational Database Rules - ACID

- **Atomicity** guarantees that each transaction is treated as a unit which either succeeds or fails completely.
- **Consistency** ensures that a transaction can only bring the database from one valid state to another.
- **Isolation** ensures that concurrent execution of transactions saves data as if each transaction was executed sequentially.
- **Durability** guarantees that once a transaction has been committed, it will remain committed even in the case of a system failure.

Database Logs Enforce the Relational Rules

- Transaction Processing
- Begin Work
- Commit Work or Rollback Work
- Database Logging Modes
- Database Logging is required for all Replication
- High Available Data Replication (HDR)

Database Transactions

- Define a unit of work that must be completed as a whole
- Ensure all work is completed
- Undo all work if any part fails
  - Begin Work - Starts the transaction
  - Commit Work - Completes the transaction (saves all work)
  - Rollback Work - Restores data to state before transaction

Example

```
 Begin Work
insert into deposit values
( TODAY, p_customer_num, p_amount );
    if ( sqlca.sqlcode != 0 ) then
        rollback work
        return
    end if
update customer set balance = (balance + p_amount)
where customer_num = p_customer_num
    if ( sqlca.sqlcode != 0 ) then
        rollback work
        return
    end if
Commit Work
    if ( sqlca.sqlcode != 0 ) then
        rollback work
        return
    end if
```

Database Logging Modes

- No Logging
- Buffered Logging
- Unbuffered Logging
- ANSI Mode Logging

No Logging

- Fastest but least safe
- Transactions are not used
- Commit or Rollback work statements not allowed

Data recovery:

- from the last checkpoint if physical log disk is available
- from last archive is physical log disk is not available

- Faster inserts, updates and deletes
- Note: Logical Logs are used for some internal activities

UnBuffered Logging

- Slowest and safest
- Transactions may be used

Data recovery:

- from last committed transaction if logical log disk is OK
- from last committed transaction on logical log tape

- All transactions are immediately written to disk (slowest performance)
- Need to monitor logs so they do not fill

Buffered Logging

- Better performance but not as safe
- Transactions may be used

Data recovery:

- from last committed transaction flushed to disk
- from last committed transaction flushed to logical log tape

- Logs stored in memory buffer until written to disk
- Need to monitor logs so they do not fill up

ANSI Mode Logging

- Enforces ANSI rules for transaction processing
- Always in a transaction (No begin work)
- Similar to Unbuffered Logging

Informix Logs

- Physical Log – image of a page before a change

- Logical Logs – record of changes in a transactions

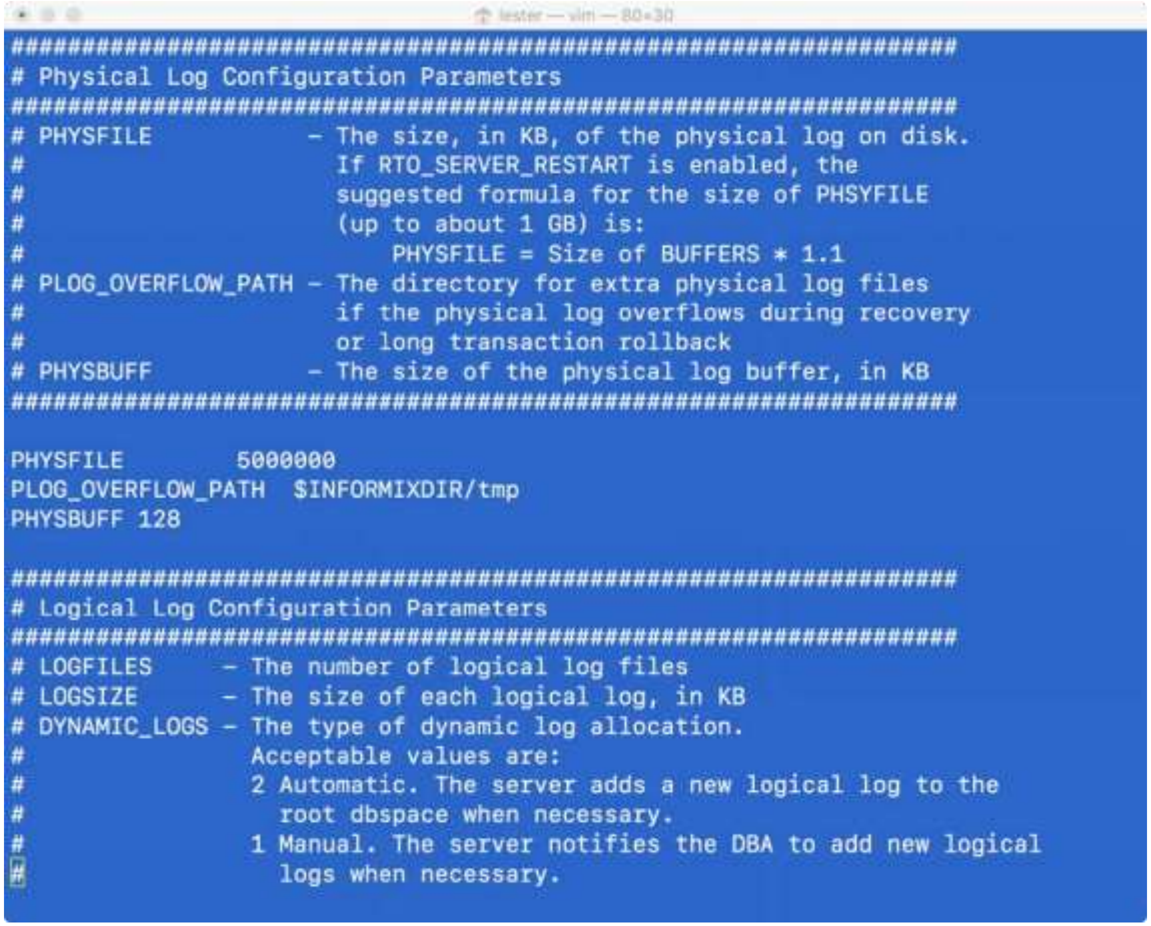
Informix Physical Log

- Stores image of disk page before any change is made to a page
- Used by Fast Recovery to return the system to the state of the last checkpoint
- A checkpoint empties the Physical Log
- When Physical Log is 75% full, Informix performs a checkpoint

Informix Logical Logs

- Informix requires a minimum of 3 Logical Logs
- The number of logs is set in ONCONFIG file
- Stores records of all changes made to a database within a transaction
- Used to rollback changes to tables
- A Logical Log is freed for reuse when it is backed up to tape and it contains no open transactions.
- ***When all logical logs are full, Server will hang***

Physical and Logical Logs: Use onparams to set after initialization



```
##################################################################
# Physical Log Configuration Parameters
##################################################################
# PHYSFILE           - The size, in KB, of the physical log on disk.
#                      If RTO_SERVER_RESTART is enabled, the
#                      suggested formula for the size of PHSYFILE
#                      (up to about 1 GB) is:
#                          PHYSFILE = Size of BUFFERS * 1.1
# PLOG_OVERFLOW_PATH - The directory for extra physical log files
#                      if the physical log overflows during recovery
#                      or long transaction rollback
# PHYSBUFF           - The size of the physical log buffer, in KB
##################################################################

PHYSFILE        5000000
PLOG_OVERFLOW_PATH  $INFORMIXDIR/tmp
PHYSBUFF 128

##################################################################
# Logical Log Configuration Parameters
##################################################################
# LOGFILES     - The number of logical log files
# LOGSIZE      - The size of each logical log, in KB
# DYNAMIC_LOGS - The type of dynamic log allocation.
#                Acceptable values are:
#                2 Automatic. The server adds a new logical log to the
#                  root dbspace when necessary.
#                1 Manual. The server notifies the DBA to add new logical
#                  logs when necessary.
```

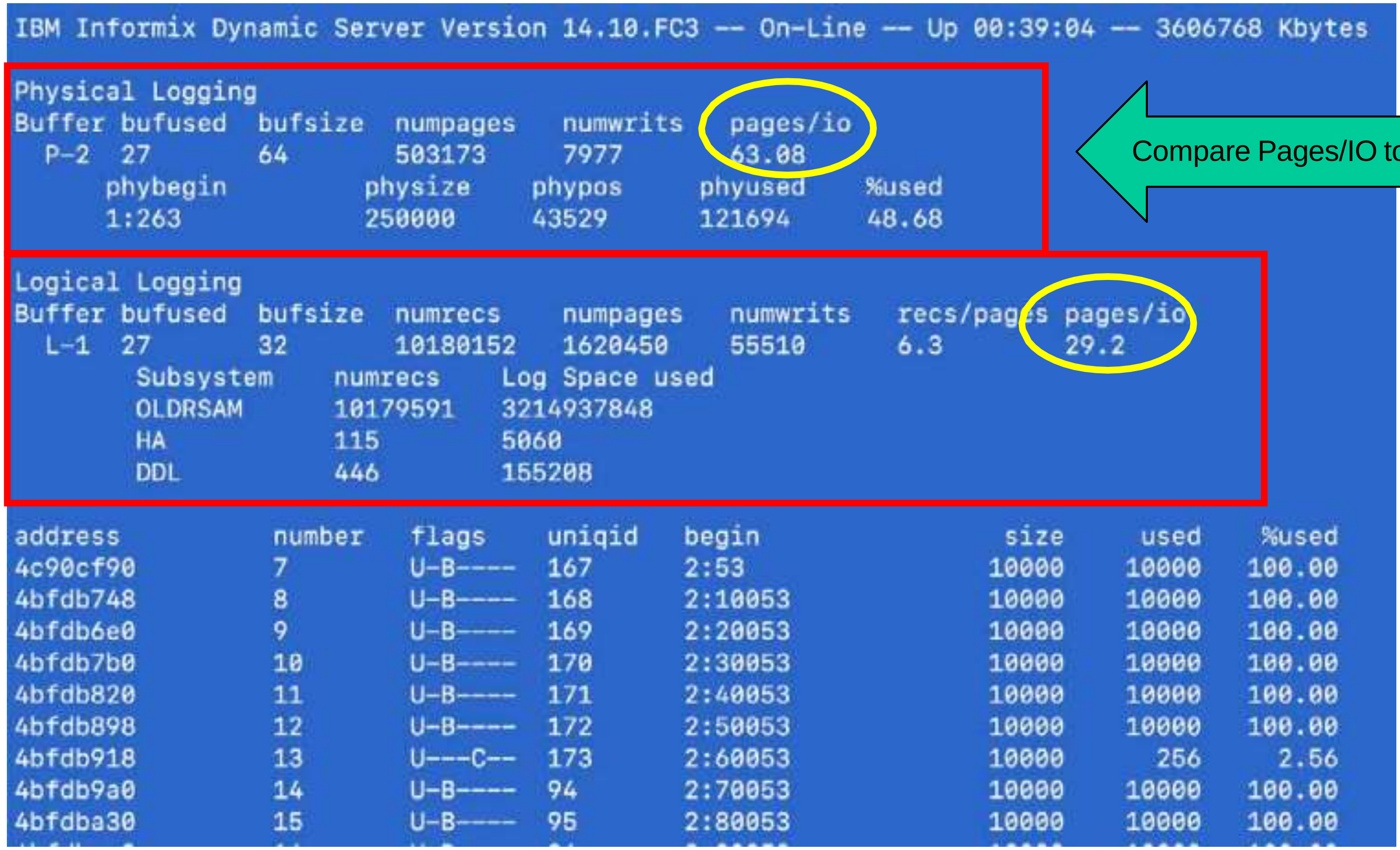
Monitoring Logical Log Status

onstat –l

- Flags:

- A - newly added
- B - Backed up
- C - Current logical log file
- F - Free, available for use
- L - Contains the last checkpoint
- U - Unreleased, in use

Onstat –l Example

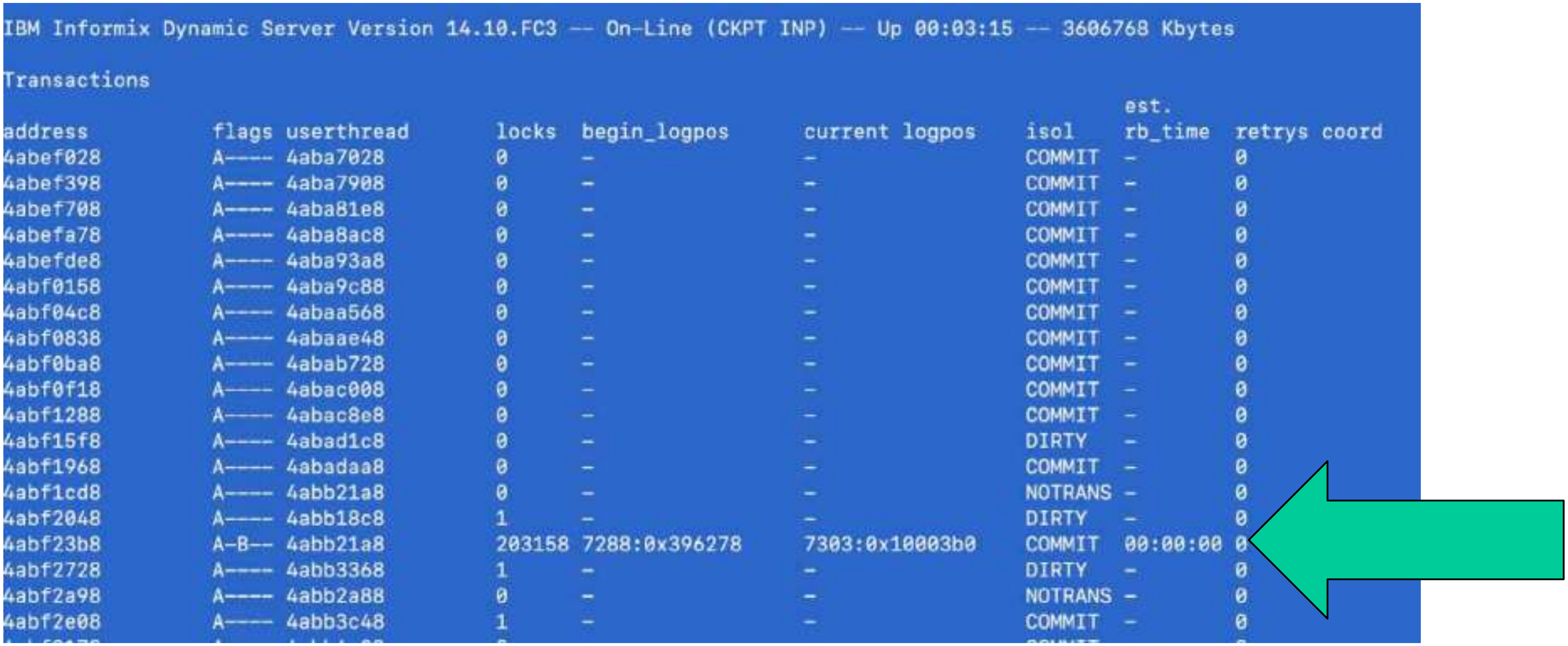


```
IBM Informix Dynamic Server Version 14.10.FC3 -- On-Line -- Up 00:39:04 -- 3606768 Kbytes

Physical Logging
Buffer bufused  bufsize  numpages   numwrits   pages/io
  P-2  27       64       503173     7977       63.08
      phybegin         physize    phypos     phyused    %used
      1:263            250000     43529      121694     48.68

Logical Logging
Buffer bufused  bufsize  numrecs    numpages   numwrits   recs/pages pages/io
  L-1  27       32       10180152   1620450    55510      6.3        29.2
        Subsystem    numrecs    Log Space used
        OLDRSAM      10179591   3214937848
        HA           115        5060
        DDL          446        155208

address          number   flags    uniqid   begin                size     used    %used
4c90cf90         7        U-B----  167      2:53                10000    10000   100.00
4bfdb748         8        U-B----  168      2:10053             10000    10000   100.00
4bfdb6e0         9        U-B----  169      2:20053             10000    10000   100.00
4bfdb7b0         10       U-B----  170      2:30053             10000    10000   100.00
4bfdb820         11       U-B----  171      2:40053             10000    10000   100.00
4bfdb898         12       U-B----  172      2:50053             10000    10000   100.00
4bfdb918         13       U---C--  173      2:60053             10000      256     2.56
4bfdb9a0         14       U-B----  94       2:70053             10000    10000   100.00
4bfdba30         15       U-B----  95       2:80053             10000    10000   100.00
```

Compare Pages/IO to Bufsize

Status of Transactions

Use: onstat -x



```
IBM Informix Dynamic Server Version 14.10.FC3 -- On-Line (CKPT INP) -- Up 00:03:15 -- 3606768 Kbytes

Transactions
                                                                                           est.
address          flags userthread       locks  begin_logpos      current logpos    isol    rb_time  retrys coord
4abef028         A----- 4aba7028        0      -                 -                 COMMIT  -        0
4abef398         A----- 4aba7908        0      -                 -                 COMMIT  -        0
4abef708         A----- 4aba81e8        0      -                 -                 COMMIT  -        0
4abefa78         A----- 4aba8ac8        0      -                 -                 COMMIT  -        0
4abefde8         A----- 4aba93a8        0      -                 -                 COMMIT  -        0
4abf0158         A----- 4aba9c88        0      -                 -                 COMMIT  -        0
4abf04c8         A----- 4abaa568        0      -                 -                 COMMIT  -        0
4abf0838         A----- 4abaae48        0      -                 -                 COMMIT  -        0
4abf0ba8         A----- 4abab728        0      -                 -                 COMMIT  -        0
4abf0f18         A----- 4abac008        0      -                 -                 COMMIT  -        0
4abf1288         A----- 4abac8e8        0      -                 -                 COMMIT  -        0
4abf15f8         A----- 4abad1c8        0      -                 -                 DIRTY   -        0
4abf1968         A----- 4abadaa8        0      -                 -                 COMMIT  -        0
4abf1cd8         A----- 4abb21a8        0      -                 -                 NOTRANS -        0
4abf2048         A----- 4abb18c8        1      -                 -                 DIRTY   -        0
4abf23b8         A-B--- 4abb21a8        203158 7288:0x396278     7303:0x10003b0    COMMIT  00:00:00 0
4abf2728         A----- 4abb3368        1      -                 -                 DIRTY   -        0
4abf2a98         A----- 4abb2a88        0      -                 -                 NOTRANS -        0
4abf2e08         A----- 4abb3c48        1      -                 -                 COMMIT  -        0
```

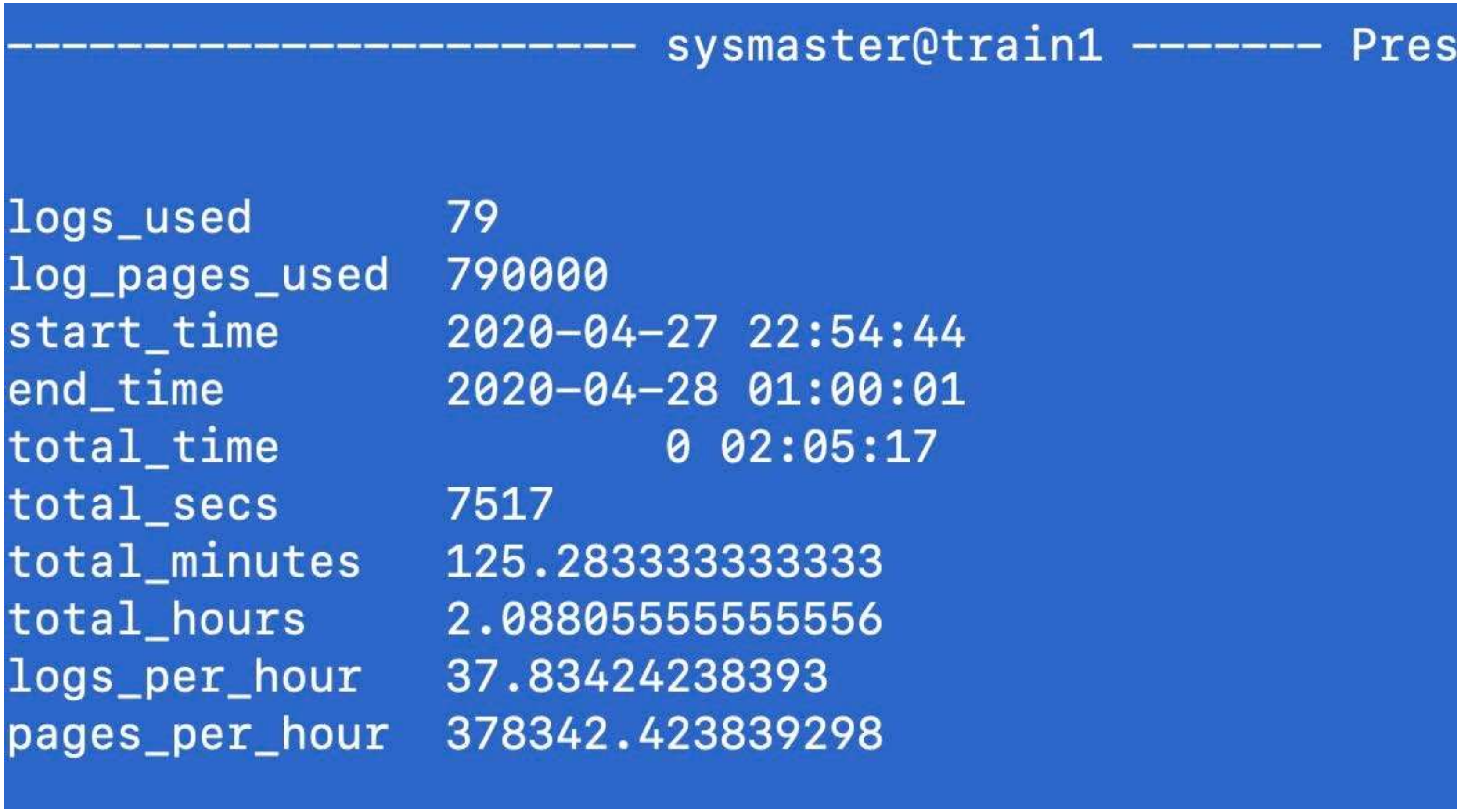
Logical Log Performance

- For HDR Planning – How much data will be going to the Secondary Servers?
- What is my Log turnover rate?
- Do I have enough Logs?
- Are the Logs too small or too big?
- Goal – Enough Logs for 4 days
- Goal – Turnover 12 to 24 Logs per hour

Logical Log Performance

```
select          count(*) logs_used,
        sum( size ) log_pages_used,
        dbinfo('utc_to_datetime', min( filltime) ) start_time,
        dbinfo('utc_to_datetime', max( filltime) ) end_time,
        (dbinfo('utc_to_datetime', max( filltime) ) - dbinfo('utc_to_datetime', min( filltime) )) total_time,
        (( max( filltime)) -  ( min( filltime) )) total_secs,
        ((( max( filltime)) -  ( min( filltime) )) /60 ) total_minutes,
        (((( max( filltime)) -  ( min( filltime) )) /60 ) /60 ) total_hours,
        ( count(*) / (((( max( filltime)) -  ( min( filltime) )) /60 ) /60 )) logs_per_hour,
        ( sum(size) / (((( max( filltime)) -  ( min( filltime) )) /60 ) /60 )) pages_per_hour
from syslogfil
where filltime > 0 ;
```

Logical Log Performance



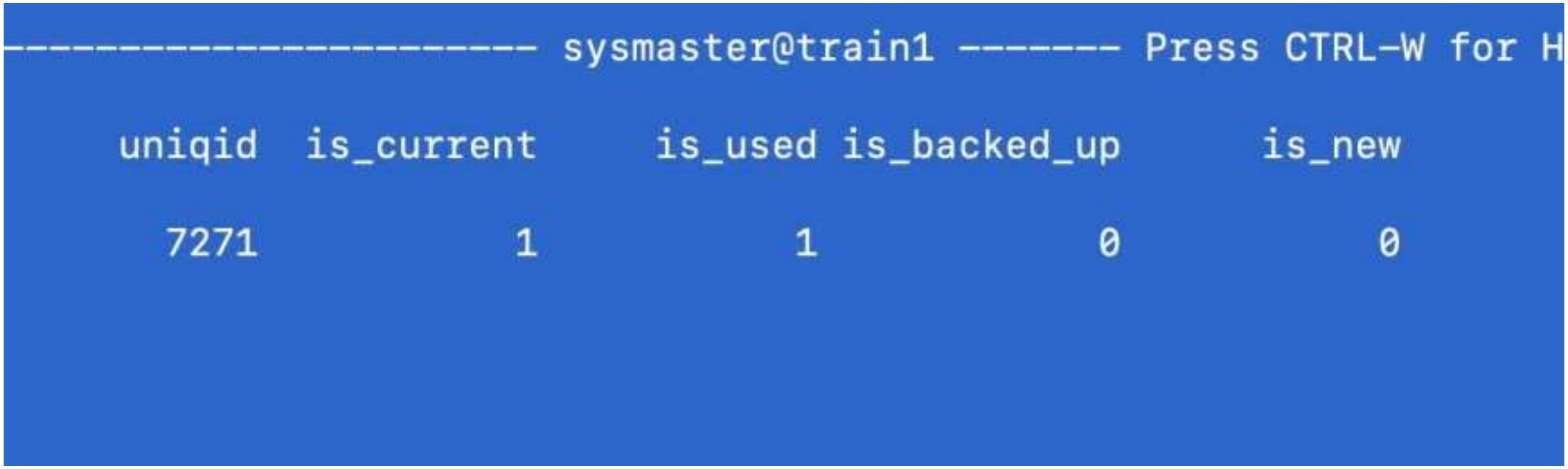
```
----------------------- sysmaster@train1 ------- Pres



logs_used        79
log_pages_used   790000
start_time       2020-04-27 22:54:44
end_time         2020-04-28 01:00:01
total_time               0 02:05:17
total_secs       7517
total_minutes    125.283333333333
total_hours      2.08805555555556
logs_per_hour    37.83424238393
pages_per_hour   378342.423839298
```

Logical Log Not Backed up

```
select uniqid, is_current, is_used, is_backed_up, is_new
from syslogs
where is_used = 1
  and is_new = 0
  and is_temp = 0
  and is_pre_dropped = 0
and is_backed_up != 1
order by uniqid;
```

Logical Log Not Backed up

Should only see the current Logical Log



```
---------------------- sysmaster@train1 ------- Press CTRL-W for H

    uniqid  is_current     is_used is_backed_up      is_new

      7271           1           1            0           0
```

Monitoring for Logical Log Failures

- Logical Log Backup Failures will result in a Blocked System
- All Logical Logs Full will result in a Blocked System
- Long Transactions will result in Transaction Rollback

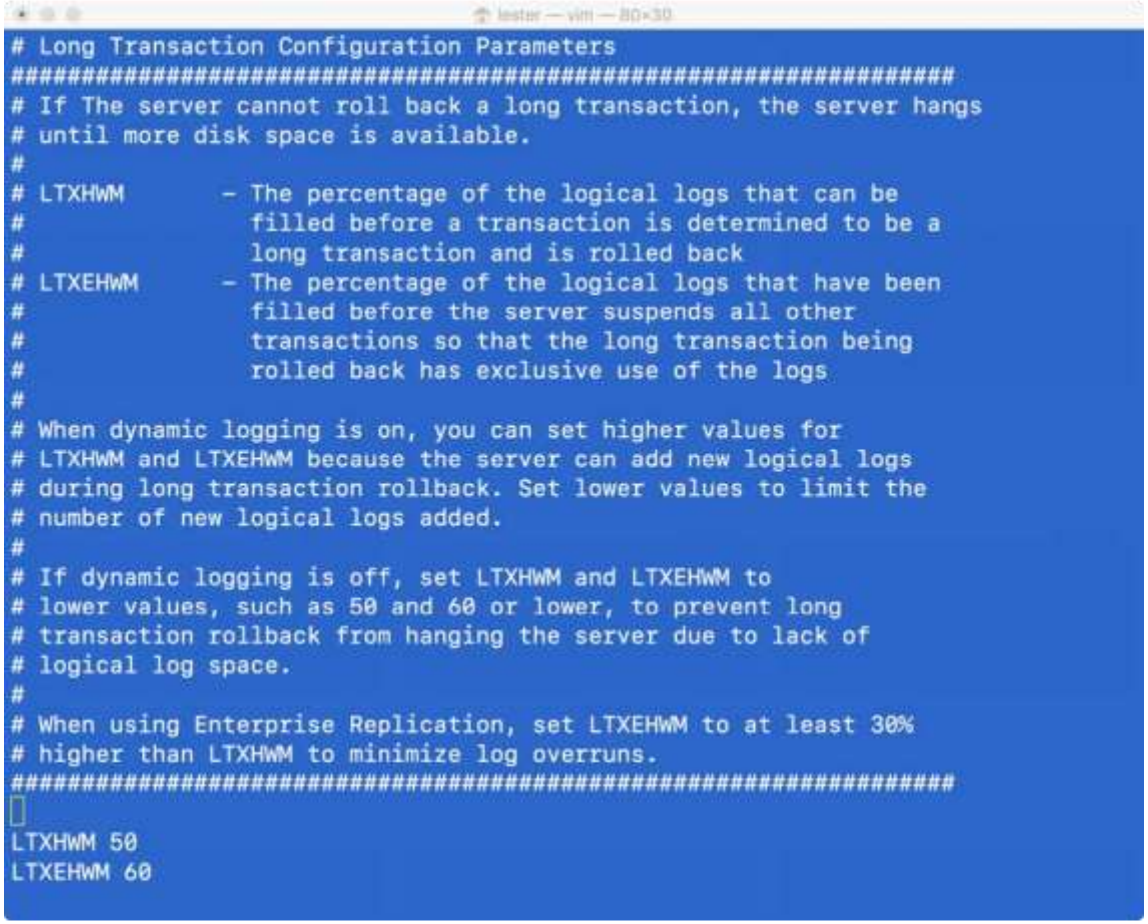
Long Transactions

- A transaction which uses too many logical logs and potentially could lock up the Informix system is a long transaction.
- Informix uses two parameters in the ONCONFIG file to define long transactions:

```
LTXHWM      50        # Long transaction high water mark percentage
LTXEHWM     60        # Long transaction high water mark (exclusive)
```

- Once a transaction uses the LTXHWM percent of logical logs it is aborted and rolled back

Long Transactions – Change default to 50%



```
# Long Transaction Configuration Parameters
###################################################################
# If The server cannot roll back a long transaction, the server hangs
# until more disk space is available.
#
# LTXHWM       - The percentage of the logical logs that can be
#                 filled before a transaction is determined to be a
#                 long transaction and is rolled back
# LTXEHWM      - The percentage of the logical logs that have been
#                 filled before the server suspends all other
#                 transactions so that the long transaction being
#                 rolled back has exclusive use of the logs
#
# When dynamic logging is on, you can set higher values for
# LTXHWM and LTXEHWM because the server can add new logical logs
# during long transaction rollback. Set lower values to limit the
# number of new logical logs added.
#
# If dynamic logging is off, set LTXHWM and LTXEHWM to
# lower values, such as 50 and 60 or lower, to prevent long
# transaction rollback from hanging the server due to lack of
# logical log space.
#
# When using Enterprise Replication, set LTXEHWM to at least 30%
# higher than LTXHWM to minimize log overruns.
###################################################################

LTXHWM 50
LTXEHWM 60
```

Informix Checkpoints

- Syncs everything in memory to disk to insure durability
- Sets Recovery point for a restore
- Clears and restarts the Physical Log

Types of Checkpoints

- Full or “sync” – all versions
- Fuzzy – (9.X – 10.X only)
- Non-Blocking Checkpoints – all versions since 11.X

Steps of a Sync Checkpoint

1. Engine blocks threads from entering “critical sections” of code
2. The page cleaner thread flushes the Physical Log buffer to log on disk
3. The page cleaner threads flush to disk all modified pages in the buffer pool (chunk write)
4. The checkpoint thread writes a checkpoint record to the Logical Log buffer
5. The Logical Log buffer is flushed to the current Logical Log file on disk
6. The Physical Log on disk is logically emptied (current entries can be overwritten)
7. The checkpoint thread updates the reserved pages with the checkpoint record information

What Causes Sync Checkpoints to Occur?

1. Physical Log becomes 75% full
2. “onmode –c” or “-ky”
3. Administrative actions (adding dbspaces, altering tables)
4. A backup or restore operation using ontape or ON-Bar
5. End of fast recovery or full recovery
6. Reuse of a Logical Log containing the oldest fuzzy operation not yet synced to disk
7. LTXHWM reached with fuzzy transactions

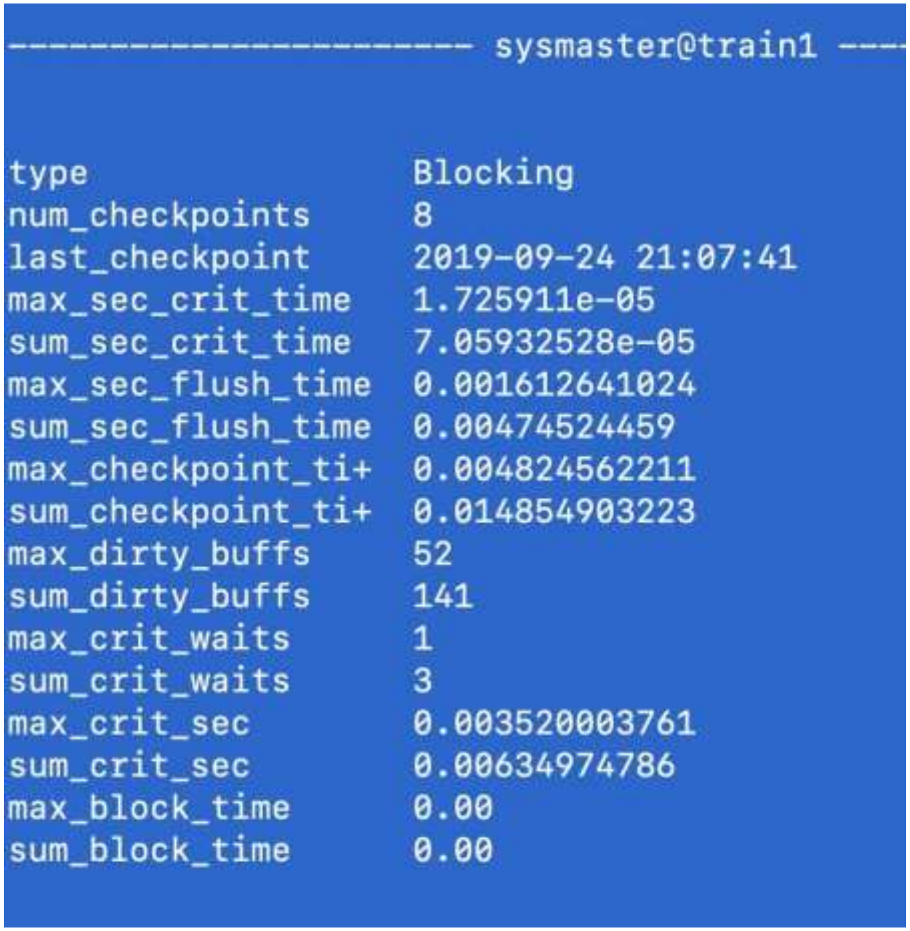
Checkpoint Performance

- What is a summary of my Checkpoint Performance?
  - Checkpoint_summary.sql
- What are the details of the last 10 Checkpoints?
  - Checkpoint_last.sql

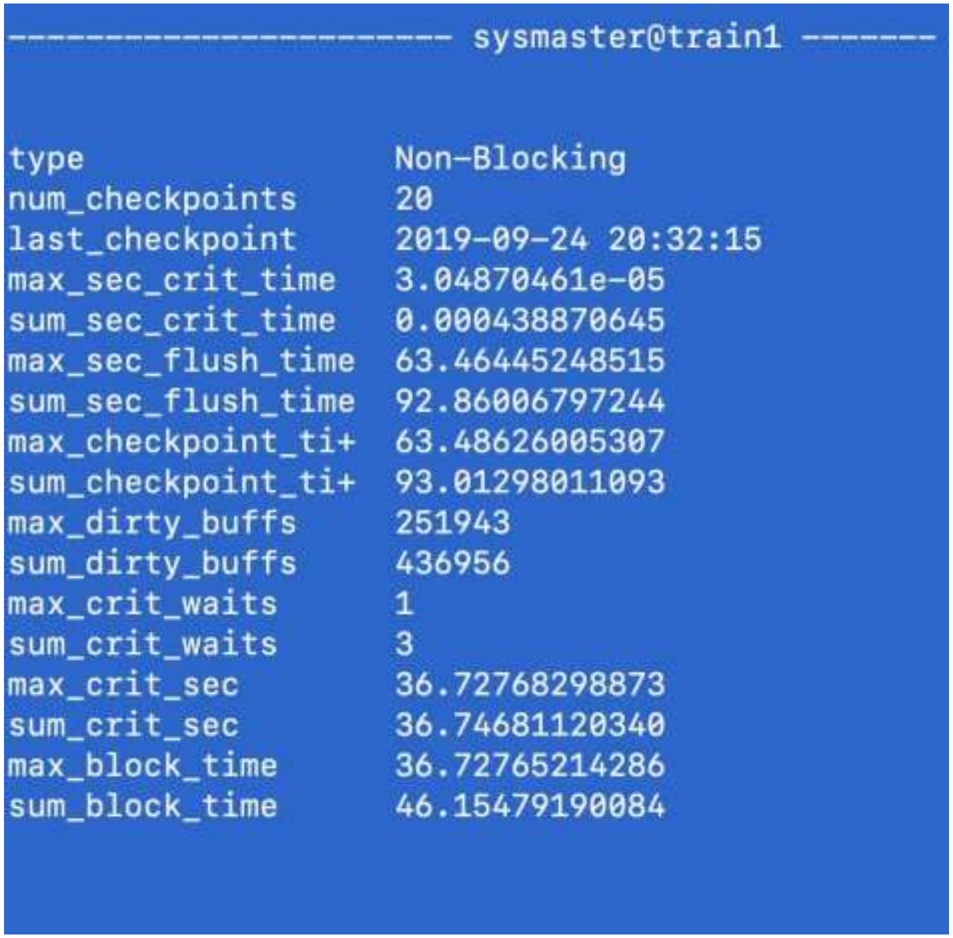
Checkpoint Performance Summary

```
select      type,
            count(*) num_checkpoints,
            max ( dbinfo( "utc_to_datetime", clock_time)) last_checkpoint,    -- Clock time of the checkpoint
            max ( crit_time ) max_sec_crit_time, -- Fractional seconds spent in critical sections
            sum ( crit_time ) sum_sec_crit_time, -- Fractional seconds spent in critical sections
            max ( flush_time ) max_sec_flush_time, -- Fractional seconds spent flushing dirty pages during the checkpoint
            sum ( flush_time ) sum_sec_flush_time, -- Fractional seconds spent flushing dirty pages during the checkpoint
            max ( cp_time ) max_checkpoint_time, -- Duration of the checkpoint in fractional seconds
            sum ( cp_time ) sum_checkpoint_time, -- Duration of the checkpoint in fractional seconds
            max ( n_dirty_buffs ) max_dirty_buffs, -- Number of dirty buffers at the beginning of the checkpoint
            sum ( n_dirty_buffs ) sum_dirty_buffs, -- Number of dirty buffers at the beginning of the checkpoint
            max ( n_crit_waits ) max_crit_waits, -- Number of processes that had to wait for the checkpoint
            sum ( n_crit_waits ) sum_crit_waits, -- Number of processes that had to wait for the checkpoint
            max ( tot_crit_wait ) max_crit_sec, -- Total time all processes waited for the checkpoint - fractional seconds
            sum ( tot_crit_wait ) sum_crit_sec, -- Total time all processes waited for the checkpoint - fractional seconds
            max ( block_time ) max_block_time, -- Longest any process had to wait for the checkpoint - fractional seconds
            sum ( block_time ) sum_block_time -- Longest any process had to wait for the checkpoint - fractional seconds
from syscheckpoint
group by 1 order by 1 ;
```

Checkpoint_summary.sql



```
------------------------ sysmaster@train1 ----


type                 Blocking
num_checkpoints      8
last_checkpoint      2019-09-24 21:07:41
max_sec_crit_time    1.725911e-05
sum_sec_crit_time    7.05932528e-05
max_sec_flush_time   0.001612641024
sum_sec_flush_time   0.00474524459
max_checkpoint_ti+   0.004824562211
sum_checkpoint_ti+   0.014854903223
max_dirty_buffs      52
sum_dirty_buffs      141
max_crit_waits       1
sum_crit_waits       3
max_crit_sec         0.003520003761
sum_crit_sec         0.00634974786
max_block_time       0.00
sum_block_time       0.00
```

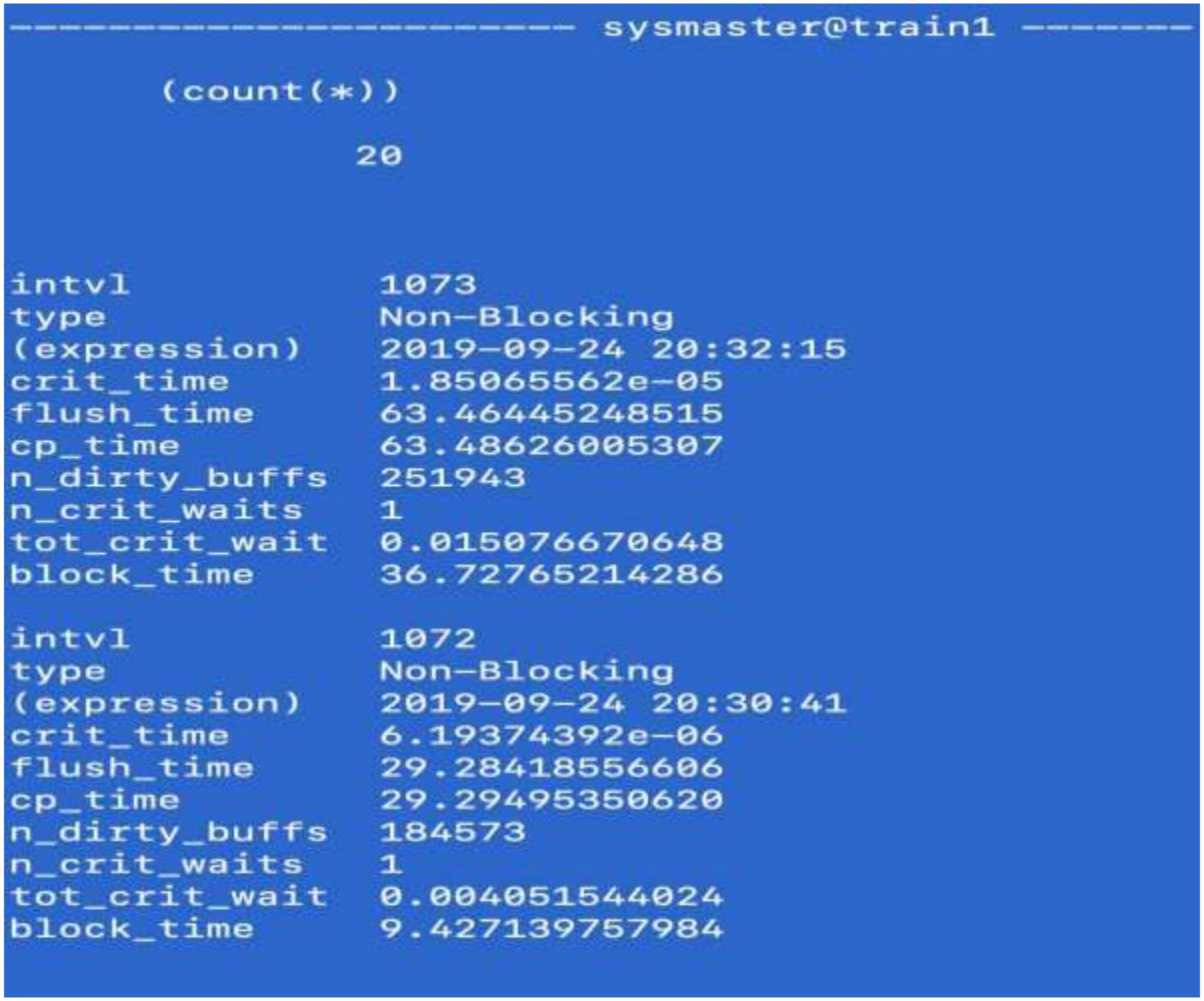


```
----------------------- sysmaster@train1 -------


type                 Non-Blocking
num_checkpoints      20
last_checkpoint      2019-09-24 20:32:15
max_sec_crit_time    3.04870461e-05
sum_sec_crit_time    0.000438870645
max_sec_flush_time   63.46445248515
sum_sec_flush_time   92.86006797244
max_checkpoint_ti+   63.48626005307
sum_checkpoint_ti+   93.01298011093
max_dirty_buffs      251943
sum_dirty_buffs      436956
max_crit_waits       1
sum_crit_waits       3
max_crit_sec         36.72768298873
sum_crit_sec         36.74681120340
max_block_time       36.72765214286
sum_block_time       46.15479190084
```

Last 10 Checkpoints

```
select first 10
        intvl,
        type,
        dbinfo( "utc_to_datetime", clock_time),   -- Clock time of the checkpoint
        crit_time, -- Fractional seconds spent in critical sections
        flush_time, -- Fractional seconds spent flushing dirty pages during the checkpoint
        cp_time, -- Duration of the checkpoint in fractional seconds
        n_dirty_buffs, -- Number of dirty buffers at the beginning of the checkpoint
        n_crit_waits, -- Number of processes that had to wait for the checkpoint
        tot_crit_wait, -- Total time all processes waited for the checkpoint - fractional seconds
        block_time -- Longest any process had to wait for the checkpoint - fractional seconds
from syscheckpoint
order by intvl desc;
```

Checkpoint_last.sql



```
----------------------- sysmaster@train1 -------

      (count(*))

              20


intvl          1073
type           Non-Blocking
(expression)   2019-09-24 20:32:15
crit_time      1.85065562e-05
flush_time     63.46445248515
cp_time        63.48626005307
n_dirty_buffs  251943
n_crit_waits   1
tot_crit_wait  0.015076670648
block_time     36.72765214286

intvl          1072
type           Non-Blocking
(expression)   2019-09-24 20:30:41
crit_time      6.19374392e-06
flush_time     29.28418556606
cp_time        29.29495350620
n_dirty_buffs  184573
n_crit_waits   1
tot_crit_wait  0.004051544024
block_time     9.427139757984
```

Backing Up Logical Logs

- Using Onbar
  - onbar –b –l

- Using Ontape
  - ontape –a
  - ontape –c

Backing Up Logical Logs

ONCONFIG entries define Log Archive Device

```
LTAPEDEV/dev/tapedev            # Log tape device path
LTAPEBLK            16                      # Log tape block size (Kbytes)
LTAPESIZE           10240                   # Max amount of data to put on log tape
(Kbytes)
```

- Setting LTAPEDEV to /dev/null
- Logs are automatically freed when no longer used
- Logs are not backed up but this does allow you to use transaction logging in your applications

Backing Up Logical Logs to /dev/null

- Setting LTAPEDEV to /dev/null
- Discards the Logical Log backups
- Cannot restore from Logical Logs
- Logs are automatically freed when no longer used
- Logs are not backed up but this does allow you to use transaction logging in your applications
- ONCONFIG entries define Log Archive Device

```
LTAPEDEV        /dev/null           # Log tape device path
```

Ontape Logical Log Backup

Continuous backup to tape - ontape –c

- Requires an operator to watch and change tapes
- Informix will hang when all the logs are full and the tape is full or stops working
- Must label tapes carefully in order to use them in a restore
- Must restart process after Informix reboots with a NEW tape
- Recommend dedicated tape drive
- Recommend creating enough logs for two days of processing in case the tape drive breaks
- Slow tape drive may impact performance

Ontape Logical Log Backup

Manual backup to tape (also called Automatic) ontape –a

- Requires an operator to start backups
- Informix will hang when all the logs are full
- Must label tapes carefully in order to use them in a restore
- Must use a NEW tape for each backup
- Slow tape drive may impact performance

Ontape Backup Logs to a Directory

- Set LTAPEDEV to a directory owned by informix and group informix
  - mkdir /informixbackups/train1/logs
  - chown informix:informix /informixbackups/train1/logs
  - chmod 770 /informixbackups/train1/logs
- User ontape to backup logs
  - ontape –a -d
- User the Alarmprogram to back up logs

Informix Fast Recovery

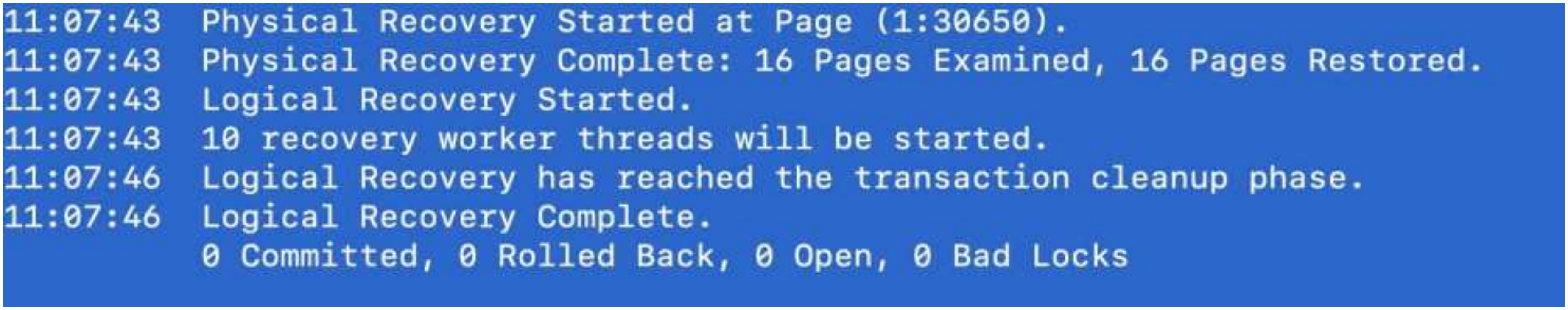
- Automatic - every time Informix is restarted
- Restores pages from Physical Log to last checkpoint
- Rolls forward all committed transactions in the Logical Log
- Rolls back all uncommitted transactions in the Logical Log
- When Informix is correctly shutdown, all transactions are flushed and Fast Recovery will have no work to do

Fast Recovery – Checkpoint

1. Physical Log data used to return all disk pages to original “synced” state (physical restore)
2. The most recent checkpoint (sync) record is located in the Logical Log files
3. All subsequent Logical Log records are rolled forward
4. Uncommitted transactions are rolled back

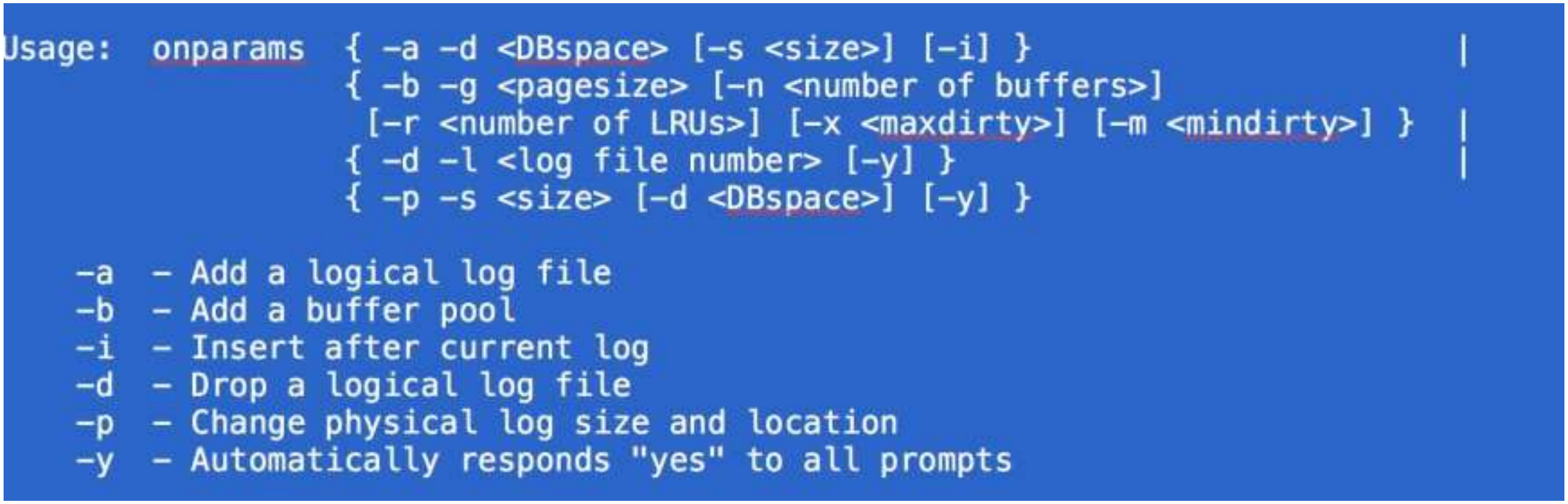
Fast Recovery

- Use onstat –m or view the logical Logs for results of Fast Recovery



```
11:07:43  Physical Recovery Started at Page (1:30650).
11:07:43  Physical Recovery Complete: 16 Pages Examined, 16 Pages Restored.
11:07:43  Logical Recovery Started.
11:07:43  10 recovery worker threads will be started.
11:07:46  Logical Recovery has reached the transaction cleanup phase.
11:07:46  Logical Recovery Complete.
          0 Committed, 0 Rolled Back, 0 Open, 0 Bad Locks
```

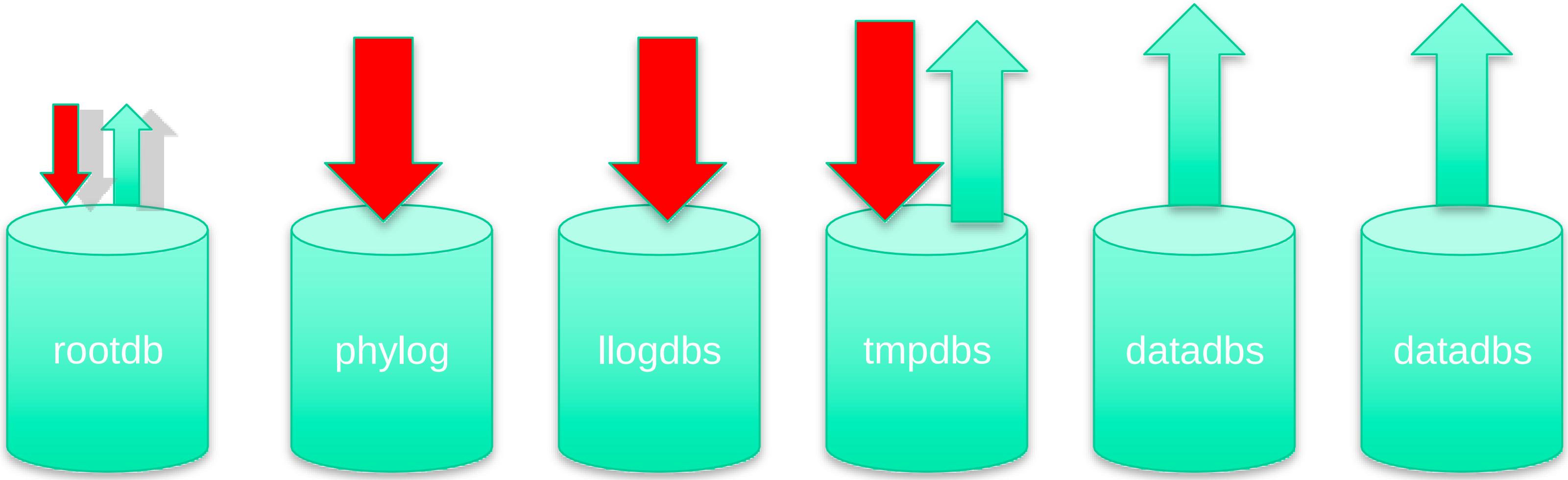
Managing Logs - Onparms



```
Usage:  onparams  { -a -d <DBspace> [-s <size>] [-i] }                     |
                  { -b -g <pagesize> [-n <number of buffers>]
                   [-r <number of LRUs>] [-x <maxdirty>] [-m <mindirty>] }  |
                  { -d -l <log file number> [-y] }                         |
                  { -p -s <size> [-d <DBspace>] [-y] }

    -a  - Add a logical log file
    -b  - Add a buffer pool
    -i  - Insert after current log
    -d  - Drop a logical log file
    -p  - Change physical log size and location
    -y  - Automatically responds "yes" to all prompts
```

Database Disk I/O

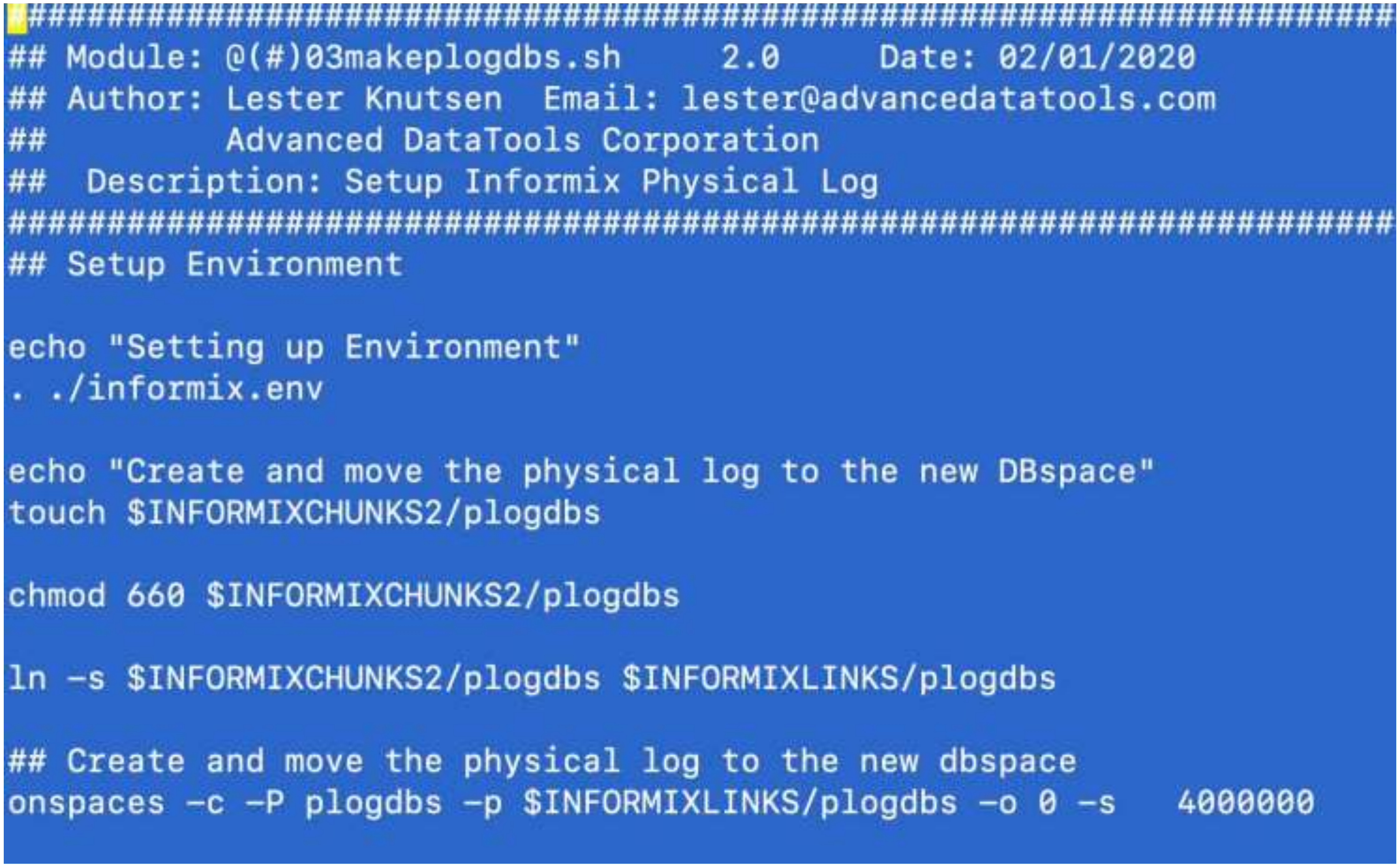
- Most Reads are from Data and Tables
- Writes will be split between Physical Log, Logical Log, Temp, and Data



Physical Log DBspace

- The Physical and Logical log will have 30-50% of all writes
- Move out of Root to separate Dbspaces
- Physical Log Size = 1.25 x Buffer Size
- A Checkpoint will occur when the Physical Log is 75% Full

Move the Physical Log to a New DBspace



```
######################################################################
## Module: @(#)03makeplogdbs.sh     2.0     Date: 02/01/2020
## Author: Lester Knutsen  Email: lester@advancedatatools.com
##         Advanced DataTools Corporation
##  Description: Setup Informix Physical Log
######################################################################
## Setup Environment

echo "Setting up Environment"
. ./informix.env

echo "Create and move the physical log to the new DBspace"
touch $INFORMIXCHUNKS2/plogdbs

chmod 660 $INFORMIXCHUNKS2/plogdbs

ln -s $INFORMIXCHUNKS2/plogdbs $INFORMIXLINKS/plogdbs

## Create and move the physical log to the new dbspace
onspaces -c -P plogdbs -p $INFORMIXLINKS/plogdbs -o 0 -s   4000000
```

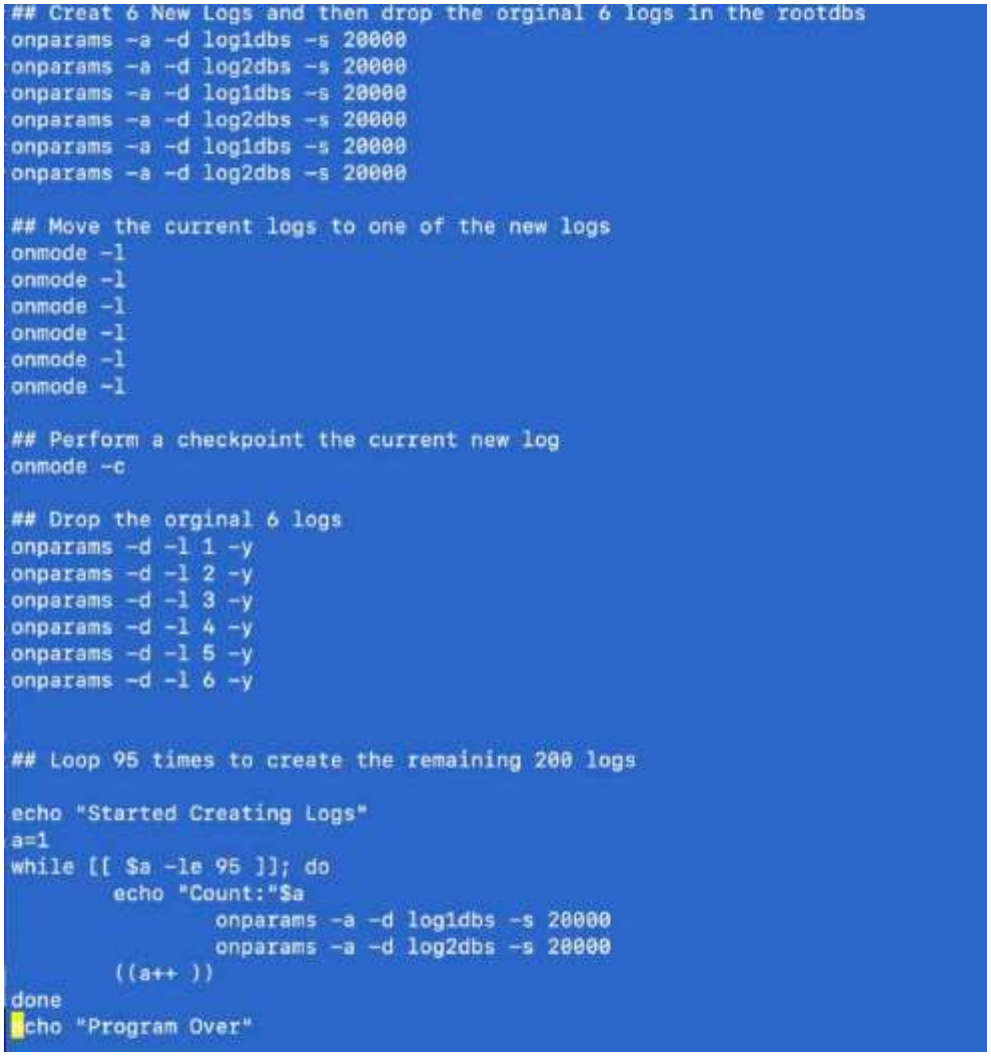
Logical Log DBspace

- The Physical and Logical log will have 30-50% of all writes
- Move out of Root to separate Dbspaces
- Logical Log Size = Hold 5-10 minutes of transactions at peak time
- Have enough Logical Logs for 4 days

Move the Logical Logs

- Create two Dbpaces for Logical Logs and alternate log location
    - One will be current and written too
    - Second will be used for backup
- Create 6 New Logs
- Make a New Log Current and with Last Checkpoint
- The Drop the old logs

Move the Logical Logs



```
## Creat 6 New Logs and then drop the orginal 6 logs in the rootdbs
onparams -a -d log1dbs -s 20000
onparams -a -d log2dbs -s 20000
onparams -a -d log1dbs -s 20000
onparams -a -d log2dbs -s 20000
onparams -a -d log1dbs -s 20000
onparams -a -d log2dbs -s 20000

## Move the current logs to one of the new logs
onmode -l
onmode -l
onmode -l
onmode -l
onmode -l
onmode -l

## Perform a checkpoint the current new log
onmode -c

## Drop the orginal 6 logs
onparams -d -l 1 -y
onparams -d -l 2 -y
onparams -d -l 3 -y
onparams -d -l 4 -y
onparams -d -l 5 -y
onparams -d -l 6 -y


## Loop 95 times to create the remaining 200 logs

echo "Started Creating Logs"
a=1
while [[ $a -le 95 ]]; do
        echo "Count:"$a
                onparams -a -d log1dbs -s 20000
                onparams -a -d log2dbs -s 20000
        ((a++ ))
done
echo "Program Over"
```

Summary and Best Practices

- Logging is Required for High Available Data Replication(HDR) and Enterprise Replication
- Check SQL Transactions for Errors and perform a Rollback
- Move the Physical Log out of Rootdbs
- Physical Log Size > 1.5 x Buffers
- Move the Logical Logs out of Rootdbs
- Logical Log Size – Enough for 4 days with an average of 6 Logs per Hour
- Setup Automatic Backup of Logical Logs